The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

Developing reinforcement learning for adaptive co-construction of continuous state and action spaces

Masato Nagayoshi[1], Hajime Murao[2], and Hisashi Tamaki[3]

[1] Niigata College of Nursing, 240, Shinnan, Joetsu 943-0147, Japan
(nagayosi@niigata-cn.ac.jp)
[2] Faculty of Cross-Cultural Studies, Kobe Univ. 1-2-1, Tsurukabuto, Nada-ku, Kobe 657-8501, Japan
(murao@i.cla.kobe-u.ac.jp)
[3] Graduate School of Engineering, Kobe University, Rokko-dai, Nada-ku, Kobe 657-8501, Japan
(tamaki@al.cs.kobe-u.ac.jp)

Abstract: Engineers and researchers are paying more attention to reinforcement learning (RL) as a key technique for realizing adaptive and autonomous decentralized systems. In general, however, it is not easy to put RL into practical use. Our approach mainly deals with the problem of designing state and action spaces. Previously, an adaptive state space construction method which is called a "state space filter" and an adaptive action space construction method which is called "switching RL," have been proposed after the other space has been fixed.
In this paper, we reconstitute these two construction methods as one method by treating the former method and the latter method as a combined method for mimicking an infant's perceptual and motor developments. Then the proposed method is based on introducing and referring to "entropy." In addition, a computational experiment was conducted using a so-called "robot navigation problem" with three-dimentional continuous state space and two-dimensional continuous action space. As a result, the validity of the proposed method has been confirmed.

Keywords: reinforcement learning, adaptive space co-construction, state space design, action space design, entropy

1 INTRODUCTION

Engineers and researchers are paying more attention to reinforcement learning (RL)[1] as a key technique in developing autonomous systems. In general, however, it is not easy to put RL to practical use. Such issues as satisfying the requirements of learning speed, resolving the perceptual aliasing problem, and designing reasonable state and action spaces for an agent, etc., must be resolved. Our approach mainly deals with the problem of designing state and action spaces. By designing suitable state and action spaces adaptively, it can be expected that the other two problems will be resolved simultaneously. Here, the problem of designing state and action spaces involves the following two requirements: (i) to keep the characteristics of the original search space as much as possible in order to seek strategies that lie close to the optimal, and (ii) to reduce the search space as much as possible in order to expedite the learning process. In general, these requirements are in conflict.

Recently an adaptive state space construction method which is called a "state space filter[3]," and an adaptive action space construction method which is called a "switching learning system[4]," have been proposed after the other space has been fixed. Here, we reconstitute these two construction methods as one method by treating them as a combined method for mimicking an infant's perceptual and motor developments The proposed method is to construct state and action spaces adaptively by introducing and referring to the "entropy" as an index of both necessity for the division of the state space in the state and sufficiency for the number of learning opportunities in the state. In addition, a computational experiment was conducted using a so-called "robot navigation problem" with three-dimentional continuous state space and two-dimensional continuous action space.

2 TYPICAL RL METHODS

2.1 Q-learning

Q-learning works by calculating the quality of a state-action combination, namely the Q-value, that gives the expected utility of performing a given action in a given state. By performing an action $a \in \boldsymbol{\mathcal{A}}_{\mathrm{Q}}$, where $\boldsymbol{\mathcal{A}}_{\mathrm{Q}} \subset \boldsymbol{\mathcal{A}}$ is the set of available actions in Q-learning and $\boldsymbol{\mathcal{A}}$ is the action space of the agent, the agent can move from state to state. Each state provides the agent with a reward r. The goal of agent is to maximize its total reward.

The Q-value is updated according to the following formula, when the agent is provided with the reward:

$$Q(s(t\text{-}1), a(t\text{-}1)) \leftarrow Q(s(t\text{-}1), a(t\text{-}1)) + \alpha_{\mathrm{Q}}\{r(t\text{-}1) + \gamma \max_{b \in \boldsymbol{\mathcal{A}}_{\mathrm{Q}}} Q(s(t), b) - Q(s(t\text{-}1), a(t\text{-}1))\} \quad (1)$$

where $Q(s(t\text{-}1), a(t\text{-}1))$ is the Q-value for the state and the action at the time step t-1, $\alpha_{\mathrm{Q}} \in [0, 1]$ is the learning rate of

Q-learning, $\gamma \in [0, 1]$ is the discount factor.

The agent selects an action according to the stochastic policy $\pi(a|s)$, which is based on the Q-value. $\pi(a|s)$ specifies the probabilities of taking each action a in each state s. Boltzmann selection, which is one of the typical actionselection methods, is used in this research. Therefore, the policy $\pi(a|s)$ is calculated as

$$\pi(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{b \in \boldsymbol{\mathcal{A}}} \exp(Q(s,b)/\tau)} \quad (2)$$

where τ is a positive parameter labeled temperature.

2.2 Actor-Critic

Actor-critic methods have a separate memory structure in order to represent the policy explicitly independently of the value function. The policy structure is called the "actor," which selects the actions, and the estimated value function is called the "critic," which criticizes the actions made by the actor. The critic is a state-value function. After each action selection, the critic evaluates the new state to determine whether things have gone better or worse than expected. That evaluation is TD-error:

$$\delta(t\text{-}1) = r(t\text{-}1) + \gamma V(s(t)) - V(s(t\text{-}1)) \quad (3)$$

where $V(s)$ is the state value.

Then, $V(s(t\text{-}1))$ is updated according to Eq. 4 in the critic based on this $\delta(t\text{-}1)$. In parallel, it is updated for the stochastic policy $\pi(a|s)$, in the actor.

$$V(s(t\text{-}1)) \leftarrow V(s(t\text{-}1)) + \alpha_{\mathrm{C}} \delta(t\text{-}1) \quad (4)$$

where $\alpha_{\mathrm{C}} \in [0, 1]$ is the learning rate of the critic.

It is typical for the normal distribution, shown in Eq. 5, to be used as the stochastic policy in the actor, when actor-critic is applied to a continuous action space[2].

In this case, both the mean $\mu(s)$ and the standard error of the mean $\sigma(s)$ about the normal distribution are calculated using TD-error $\delta(t\text{-}1)$ in the actor, as in Eqs. 6 and 7.

$$\pi(a|s) = \frac{1}{\sigma(s)\sqrt{2\pi}} \exp\left(\frac{-(a-\mu(s))^2}{2\sigma(s)^2}\right) \quad (5)$$

$$\mu(s(t\text{-}1)) \leftarrow \mu(s(t\text{-}1)) + \alpha_{\mu} \delta(t\text{-}1)(a(t\text{-}1) - \mu(s(t\text{-}1))) \quad (6)$$

$$\sigma(s(t\text{-}1)) \leftarrow \sigma(s(t\text{-}1)) + \alpha_{\sigma} \delta(t\text{-}1)((a(t\text{-}1) - \mu(s(t\text{-}1)))^2 - \sigma(s(t\text{-}1))^2) \quad (7)$$

where $\alpha_{\mu} \in [0, 1], \alpha_{\sigma} \in [0, 1]$ are the learning rate of the mean and the standard error of the mean, respectively. Here, if Eq. 7 is used directly, the standard error could be 0 or a negative value. Therefore it is necessary when setting the standard error to be creative and to specify the range, etc.

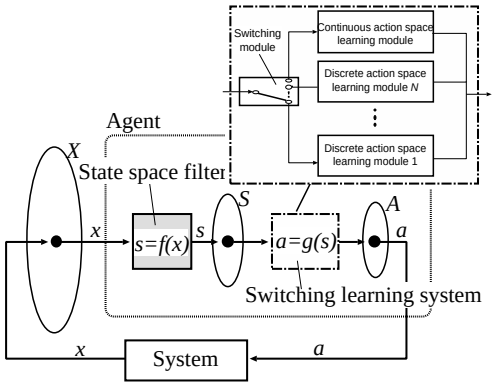

Fig. 1. Proposed development RL model

3 DEVELOPING RL

3.1 Outline of a Computational Model

In this section, we propose developing an RL model to mimic the processes of an infant's perceptual and motor developments. The proposed model is constructed by "state space filter[3]" to mimic the process of perceptual development in which perceptual differentiation progresses as the infant becomes older and more experienced, and a "switching learning system[4]" to mimic the process of motor development in which gross motor skills develop before fine motor skills, as shown in Fig. 1.

This model mimics the process of perceptual development by differentiating the state space gradually from the undifferentiated state space. In parallel, this model also mimics the process of motor development by switching discrete action space learning modules (hereafter called "DA modules") from a more coarse-grained DA module to a more fine-grained DA module, and finally switching to a continuous action space learning module (hereafter called a "CA module").

3.2 State and Action Spaces Construction Method

3.2.1 Basic Idea

A variety of methods can be considered to acquire the state space filter and the switching learning module. Here, we propose a method based on introducing and referring to the entropy, which is defined by action selection probability distributions in a state, and the number of learning opportunities in the state. It is expected that the proposed method (i) is able to learn in parallel the state space filter and the switching learning system, and (ii) does not required specific RL methods for the learning module.

The entropy of action selection probability distributions using Boltzmann selection in a state $H_{\mathrm{D}}(s)$ is defined by

$$H(s) = -(1/\log|\boldsymbol{\mathcal{A}}_{\mathrm{D}}|) \sum_{a \in \boldsymbol{\mathcal{A}}_{\mathrm{D}}} \pi(a|s) \log \pi(a|s) \quad (8)$$

where $\boldsymbol{\mathcal{A}}_{\mathrm{D}}$ is the action space and $|\boldsymbol{\mathcal{A}}_{\mathrm{D}}|$ is the number of available actions of the DA module.

The state space filter is adjusted and the learning module is switched by treating this entropy $H(s)$ as an index of the

necessity of division for an inner state s and the action space. In parallel, the learning module is switched by treating this entropy $H(s)$ as an index of sufficiency for the number of learning opportunities in the state.

If the entropy does not get smaller despite the learning module having had a sufficient number of learning opportunities in the inner state, then the state space filter is adjusted by dividing the inner state, and the learning module is switched to a more fine-grained one. In contrast, if the entropy gets small regardless of the number of learning opportunities, the learning module is switched to the CA module because the number of learning opportunities is sufficient.

In this article, Q-learning and actor-critic are applied to the DA module and the CA module, respectively. The learning module is switched in the order of Q-learning with an action space divided evenly into $n, 2n, \cdots, 2^{(N-1)}n$, and finally ending with actor-critic, where N is the number of DA modules.

3.2.2 Adjustment of State Space Filter

If $L(s) > \theta_{\mathrm{L}}$ and $H(s) > \theta_{\mathrm{H}}$, where $L(s)$ is the number of learning opportunities in s, θ_L is a threshold value of the number of learning opportunities, θ_{H} is a threshold value of the entropy, and θ_{L} is set at a sufficiently large number, then the state space filter is adjusted by dividing the range of the input state mapped to the inner state s into two parts for each dimension, and mapping each part to a different inner state. Simultaneously, the learning module is switched. Through this operation, the size of the inner state space increases by $(2^M - 1)$ after being divided, where M is the number of dimensions. Also note that the values of the new $2M$ inner states are the value of the inner state before it was divided.

In addition, after the learning module is switched to the CA module, if $L(s) > \theta_{\mathrm{L}}$, then the state space filter is adjusted by dividing the inner state so that it is more fine-grained.

3.2.3 Switching of Learning Module

If $H(s) > \theta_{\mathrm{H}}$, then the learning module is switched to the CA module because the number of learning opportunities is then sufficient. In the procedure to switch controllers, the result of Q-learning is succeeded by actor-critic and the following procedure is carried out. 1. The state value of the critic, $V(s)$, is initialized by

$$V(s) = \sum_{a \in \boldsymbol{\mathcal{A}}_{\mathrm{Q}}} \pi(a|s) \cdot Q(s,a) \tag{9}$$

2. The normal probability distribution used by the actor is calculated by

$$\mu(s) = \arg\max_{a \in \boldsymbol{\mathcal{A}}_{\mathrm{Q}}} Q(s,a), \tag{10}$$

$$\sigma(s) = |A_{\mathrm{Q}}(\arg\max_{a \in \boldsymbol{\mathcal{A}}_{\mathrm{Q}}} Q(s,a))|/6 \tag{11}$$

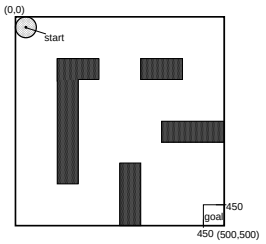

Fig. 2. Robot navigation problem

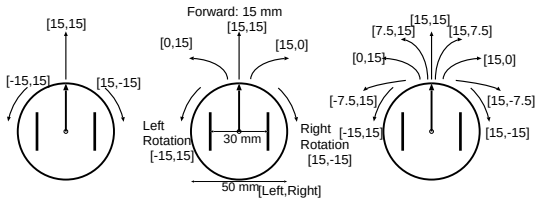

Fig. 3. Available actions of 3 Q-learning methods

where $|A_{\mathrm{Q}}(i)|$ is the range of the action space which represents action i of Q-learning.

If $L(s) > \theta_L$ and $H(s) > \theta_{\mathrm{H}}$, then the learning module is switched to more fine-grained DA module, and finally ends with the CA module. Simultaneously, the state space filter is adjusted.

The Q-values of newly added actions a_i at this time are set according to the following formula :

$$Q(s,i) = \max_{j \in i-1, i+1} Q(s,j) \tag{12}$$

where action $i-1$ and $i+1$ are adjacent to action i. This formula is set in consideration of a more efficient search as well as the idea of the optimistic initial values.

4 COMPUTATIONAL EXAMPLE

4.1 Robot Navigation Problem

The proposed method is applied to a so-called "robot navigation problem" navigating a learning agent from a starting point to a goal area in a continuous state space, as shown in Fig. 2.

Here, the agent has a circular shape (diameter 50 mm), and the continuous space is 500 mm$\times 500$ mm bounded by the external wall, with internal walls as shown in black. The agent can observe the center of its own position and its own direction: $(x_{\mathrm{A}}, y_{\mathrm{A}}, \theta_{\mathrm{A}})$ as the input. The agent has two wheels and can move in any direction, i.e., it can decide right-and-left wheel rotation rates (mm/step): $(\omega_{\mathrm{L}} \in [-15.0, 15.0], \omega_{\mathrm{R}} \in [-15.0, 15.0])$ as the output.

The positive reinforcement signal $r_t = 10$ (reward) is given to the agent only when the center of the agent arrives in the goal area, and the reinforcement signal is $r_t = 0$ at all other steps. The period from when the agent is located at the starting point to when the agent is given a reward, or $100,000$ steps pass away, labeled as 1 episode, is repeated.

4.2 Comparison with adaptive methods

We have confirmed that a combined method of state space filter and switching learning system (hereafter called the "FS" method) shows a better performance than three Q-learning methods where the number of actions is designed to be 3, 5, and 9 (hereafter called the "Q3," "Q5," and "Q9" methods, respectively), and an actor-critic method (hereafter called the "AC" method)with the state space divided evenly into $5 \times 5 \times 8$, $10 \times 10 \times 16$, and $20 \times 20 \times 32$ in this task. Fig. 3 shows the wheel rotation rates on each actions of three Q-learning methods.

In this section, the FS method is compared with three methods using the switching learning system with the state space divided evenly into $5 \times 5 \times 8$, $10 \times 10 \times 16$, and $20 \times 20 \times 32$ (hereafter called the "S5," "S10," and "S20" methods, respectively), and three Q-learning methods, the Q3, Q5, and Q9 methods, using the state space filter (hereafter called the "FQ3," "FQ5," and "FQ9," methods, respectively). Here, an initial state space filter is designed that divides the state space evenly into $5 \times 5 \times 8$ spaces, and the switching learning system is constructed by four learning modules: the "Q3," "Q5," "Q9," and "AC" methods.

All initial Q-values are set at 5.0 as the optimistic initial values[1] for Q-learning methods and the range of $\sigma(x)$ is set at $[0.001, 30.0]$ for the AC method. Here, the maximum limit of $\sigma(x)$ is set so that it becomes the size of the action space: 30.0. Further, the adjustment of the state space filter is assumed until the third attempt in all inner states because it is impossible to evaluate sufficiency for division of the state space.

Computer experiments have been carried out with the parameters shown in Table 1. Here, θ_{H} was set to about 0.359, the maximal value of the entropy when the highest selection probability for one action is 0.9, θ_{L} was set to a large enough number.

The average number of steps required to accomplish the task was observed during learning over 20 simulations with different methods, as described in Fig. 4. The average size of the inner state space was observed during learning over 20 simulations with different methods, as described in Fig. 5.

It can be seen from Fig. 4 that, (1) the FS method showed a worse performance than the FQ3, FQ5, FQ9 and S10 methods with regard to the learning speed, but a better performance than any other methods with regard to the control rule obtained. (2) the S5 method couldn't acquire any proper control rule.

Table 1. Parameters for experiments

Parameter	Value	Parameter	Value
$\alpha_{\mathrm{Q}}, \alpha_{\mathrm{C}}, \alpha_{\mu}, \alpha_{\sigma}$	0.1	γ	0.9
θ_{H}	0.4	τ	0.1
θ_{L}	1000		

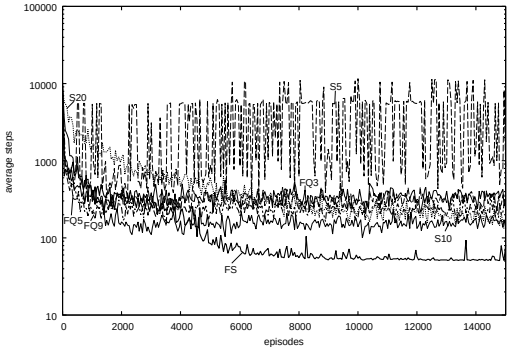

Fig. 4. Required steps

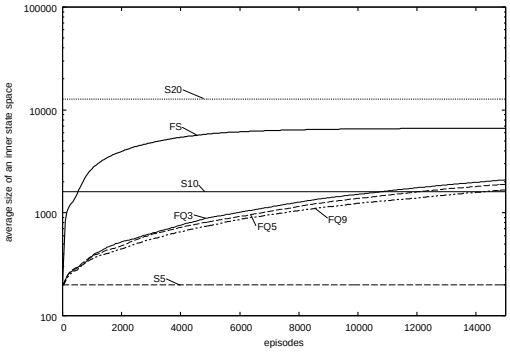

Fig. 5. Size of the inner state space

It can be seen from Fig. 5 that, (1) the FS method is smaller than the S20 method, but larger than any other method except the S20, (2) and is not growing since about 5,000 episode with regard to the size of the inner state space.

Therefore, we have confirmed that the FS method demonstrates better performances than any other method for the robot navigation problem.

5 CONCLUSION

In order to design suitable state and action spaces adaptively, we have proposed the developing RL model, and the state and action spaces co-construction method referring to "entropy." Then, with a computational experiment we confirmed that the combined method of the state space filter and the switching learning system shows better performances than any other method for the robot navigation problem with continuous state and action space.

Our future projects include considering more complicated problems and real-world problems, etc.

REFERENCES

[1] R.S. Sutton and A.G. Barto (1998), Reinforcement Learning, A Bradford Book, MIT Press.

[2] H. Kimura and S. Kobayashi (2000), An Analysis of Actor-Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imperfect Value Functions, JSAI Jour., **15**(2), 267-275 (in Japanese).

[3] M. Nagayoshi, H. Murao, and H. Tamaki (2006), A State Space Filter for Reinforcement Learning, Proc. AROB 11th'06, 615-618(GS1-3).

[4] M. Nagayoshi, H. Murao and H. Tamaki (2010), A Reinforcement Learning with Switching Controllers for Continuous Action Space, Artifi cial Life and Robotics,**15**, 97-100.